Journal of Computer and System Sciences ••• (••••) •••–•••

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

Exploring gate-limited analytical models for high-performance network storage servers

Glenford Mapp *, Dhawal Thakker, Orhan Gemikonakli

School of Engineering and Information Sciences, Middlesex University, Hendon Campus, London, UK, NW4 4BT

ARTICLE INFO

Article history:
Received 20 January 2010
Received in revised form 7 May 2010
Accepted 9 August 2010
Available online xxxx

Keywords:
Gate-limited service
Markov models
Prefetching

ABSTRACT

Gate-limited service is a type of service discipline found in queueing theory and can be used to describe a number of operational environments, for example, large transport systems such as buses, trains or taxis, etc. Recently, there has been the observation that such systems can also be used to describe interactive Internet Services which use a Client/Server interaction. In addition, new services of this genre are being developed for the local area. One such service is a Network Memory Server (NMS) being developed here at Middlesex University. Though there are several examples of real systems that can be modelled using gate-limited service, it is fair to say that the analytical models which have been developed for gate-limited systems have been difficult to use, requiring many iterations before practical results can be generated. In this paper, a detailed gate-limited bulk service queueing model based on Markov chains is explored and a numerical solution is demonstrated for simple scenarios. Quantitative results are presented and compared with a mathematical simulation. The analysis is used to develop an algorithm based on the concept of optimum operational points. The algorithm is then employed to build a high-performance server which is capable of balancing the need to prefetch for streaming applications while promptly satisfying demand misses. The algorithm is further tested using a systems simulation and then incorporated into an Experimental File System (EFS) which showed that the algorithm can be used in a real networking environment.

1. Introduction

The analysis of queueing systems has a long and interesting history. The results of this research have been used in many practical systems, from optimal queueing strategies to serve customers at banking centres to token ring systems in high-speed computer networks. These systems are studied to yield some important results such as the average waiting time for a customer in the system or the average number of customers in the system as well as to address general performance issues such as how systems respond under load, etc. Queueing systems can be defined by various parameters including the arrival time of the customers, the service time, and the number of servers. These parameters are used to define key systems such as the M/M/1 queue which is a single queue, served by a single server, in which the arrival rate is Poisson and the departure rate is exponentially distributed. The M/M/1 queue has been extensively studied.

Traditionally, two types of service disciplines and their variants have been studied. The first is called **exhaustive** service where the server serves everyone in the queue until the queue is empty. This means that customers arriving at the queue while the server is at the queue are served during the current service period [1]. The second is called **gated** service, where the server only serves the customers it finds in the queue at the beginning of the service period for that queue. Customers

* Corresponding author.
E-mail addresses: g.mapp@mdx.ac.uk (G. Mapp), d.thakker@mdx.ac.uk (D. Thakker), o.gemikonakli@mdx.ac.uk (O. Gemikonakli).

0022-0000/$ – see front matter © 2010 Published by Elsevier Inc.
doi:10.1016/j.jcss.2010.08.002

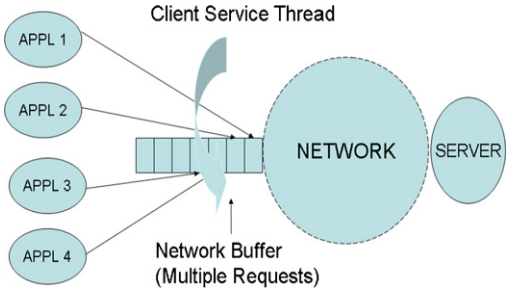

Fig. 1. A example of gate-limited multiserver service.

arriving during the current service period must therefore be serviced during subsequent server visits to that queue. In exhaustive-limited and gate-limited systems, the relevant discipline is followed but only a maximum number of customers, denoted by K, are served in any one visit to the queue. Though we have exact solutions for exhaustive and gated service [2,3], solutions for exhaustive-limited and gate-limited system have been more difficult to obtain, with numerical solutions requiring large amounts of computational power as the number of queues increase. Recent efforts have therefore been focused on getting faster algorithms to compute the waiting times in these systems.

The study of multiserver systems, in which a number of servers simultaneously serve the same queue, intensified within the computer community with the development of multiprocessors systems in the 90s [4]. With the advent of mobile communications [5], multiserver systems have become more closely studied. For multiserver systems with a limited number of servers, say K servers, the deployment of servers can also be classified as exhaustive-limited and gate-limited. In exhaustive-service, all K servers are always deployed and so a customer can enter service at an empty server while other customers are being served by other servers. This is also called **Partial Batch Service**. An example of this kind of service is a wireless network such as 3G [6] and other mobile systems, where channels are allocated in a dynamic manner.

In gate-limited service, if the number of customers in the queue at the end of the service period is less than the maximum K, say m, then only m servers are assigned for this service period. Customers arriving after service has begun must be served in subsequent cycles. There are many examples of real systems which could be modelled as gate-limited systems including bulk transport systems such as buses, trains, etc. However, with the emergence of the Internet, distributed systems which use Client/Server semantics can also be modelled as a gate-limited bulk service. An example of such a system is a network-based service in which applications can post requests using a network buffer. The buffer is then sent over to the server to satisfy these requests. New requests that arrive after the buffer has been sent, must wait until replies to previous requests have been returned to the client machine, hence this can be regarded as gate-limited multiserver system. This arrangement is illustrated in Fig. 1.

With the development of faster Local Area Networks (LANs), services that were mostly done on the local machine can now be done over the network. One area where there has recently been a lot of research is in the area of storage using the memory on other machines on a local area network to store data such as the Network Memory Server (NMS) here at Middlesex University. This effort has been focused on looking at how a server such as the NMS can be used to provide a high-performance service such that streaming applications can stream data without stalling while promptly satisfying demand misses to allow programs to continue their execution. Such a system can be shown to be a gate-limited system but with the additional requirement of determining the amount of prefetching and the number of demand misses which must be satisfied in a given cycle.

In order to investigate this further, it was first decided to develop a new model for gate-service based on Markov chains. The analytical model was then used to develop an algorithm to balance prefetching and demand paging. This algorithm was then tested using a simulation at the systems level. Finally, an Experimental File System (EFS) was developed to test the algorithm on a real system. The rest of the paper is structured as follows: Section 2 examines previous approaches, while Section 3 introduces a new analytical model based on Markov chains. In Section 4 the approach taken to arrive at a mathematical solution is outlined. In Section 5, the issues involved in designing a high-performance server are investigated while Section 6 details the development and testing of an autonomous algorithm. Finally conclusions are discussed in Section 7.

2. Related work

2.1. Related work on gate-limited service

There have been several approaches used to analyse these operational environments [7]. Cyclic systems with probabilistically-limited service were examined in [8]. The authors in [9], explored pseudo-conservation laws to examine cyclic systems with several limited service policies, including gate-limited ones. In both cases, solutions were found but several iterations were required to yield a useable result. Recent work explored new algorithms to improve the accuracy

Please cite this article in press as: G. Mapp et al., Exploring gate-limited analytical models for high-performance network storage servers, J. Comput. System Sci. (2010), doi:10.1016/j.jcss.2010.08.002

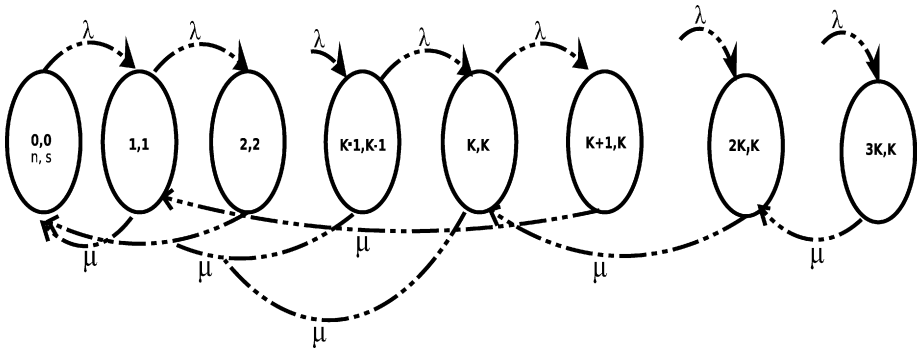

Fig. 2. Partial Batch Model.

and speed with moderate success [10,11]. However, it is fair to say that practical numeric solutions for gate-limited service remain fairly rare. In that regard, we think this paper and its explicit use of Markov chains may point to a new approach.

2.2. Multiserver exhaustive-limited service (Partial Batch Model)

We first examine the multiserver exhaustive-limited service or the Partial Batch Model (PBM) described in [12]. In this model a server can serve up to a maximum of K requests. If there are less than K requests in the system, the server begins to serve these requests. Furthermore, when there are less than K requests being serviced, new arrivals enter service until K requests are served or the queue is empty. The amount of time required to service requests is an exponentially distributed random variable with mean $\frac{1}{\mu}$.

This model is represented in Fig. 2. Each state of the model is represented in terms of n and s where n is the total number of requests in the system and s is the number of requests being served. It can be seen from the figure that any new arrival enters service immediately as long as there are less than K requests being served. The time taken to service those requests is exponentially distributed to a mean value of $\frac{1}{\mu}$.

A stochastic balance equation for the model can be written as:

$$0 = -(\lambda + \mu)p_n + \mu p_{n+k} + \lambda p_{n-1} \tag{1}$$

$$0 = -\lambda p_0 + \mu p_1 + \mu p_2 + \cdots + \mu p_{K-1} + \mu p_K \tag{2}$$

Eq. (1) can be rewritten as:

$$\left[\mu D^{K+1} - (\lambda + \mu)D + \lambda\right]p_n = 0 \tag{3}$$

where p_n represents the state probability and $n = 0, 1, 2, \ldots$ etc.

By finding the root r_0 of this equation that is between 0 and 1, one can work out the mean queue length (L) and average waiting time (W) for the queue, using the equations below:

$$L = \frac{r_0}{1 - r_0} \quad \text{and} \quad W = \frac{r_0}{\lambda(1 - r_0)} \tag{4}$$

Note that for $K = 1$, we have the results for the M/M/1 queue, with $r_0 = \rho = \frac{\lambda}{\mu}$. More detailed analysis is also discussed in [13,14].

3. A new multiserver gate-limited model

We now present a Markov model for gate-limited multisever service. The model is based on Markov states represented by the same two parameters used in the Partial Batch Model, n and s. For each cycle, we serve a maximum of K requests, so $s_{\max} = K$. The model is shown in Fig. 3. We start off with the empty state $(0, 0)$. If a request arrives, the system moves to state $(1, 1)$ as the request is immediately sent to the server. If more requests arrive before the server has returned, they are not served until the server returns. So we can represent this by the first chain in our model. When the first request is served, the number of people served in the next cycle will depend on the number of requests in the queue when the service time has been completed. Thus for high instantaneous arrival rates, the system moves up the chains and for lower instantaneous arrival rates, it descends the chains.

Please cite this article in press as: G. Mapp et al., Exploring gate-limited analytical models for high-performance network storage servers, J. Comput. System Sci. (2010), doi:10.1016/j.jcss.2010.08.002

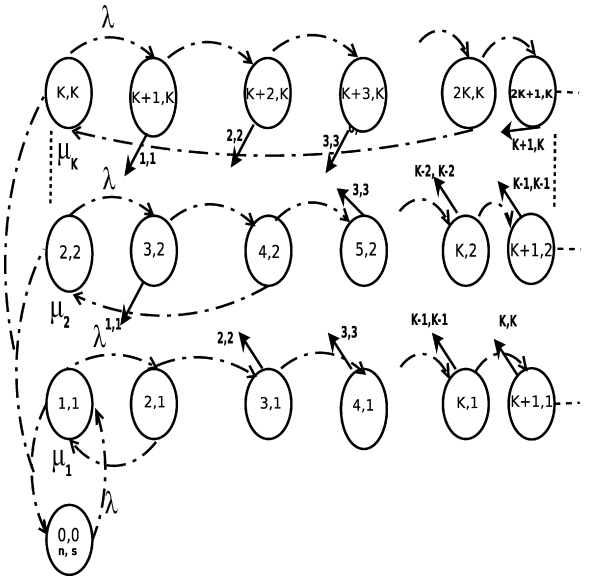

Fig. 3. Markov chains for gate-limited multiserver model.

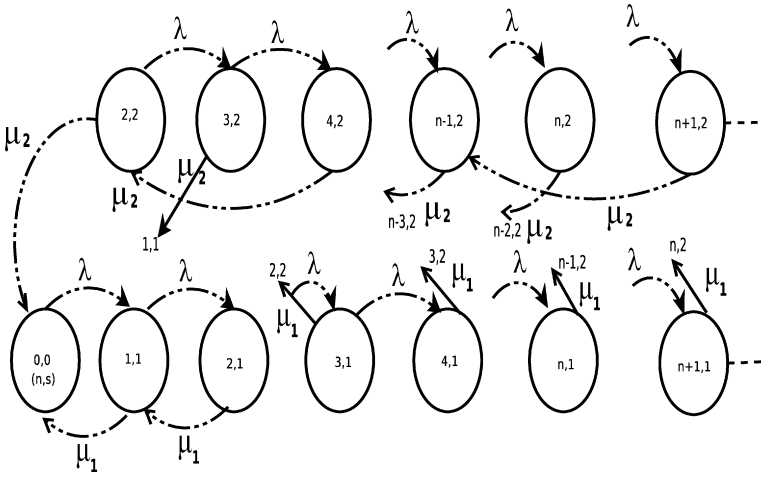

Fig. 4. Simple gate-limited model with $K = 2$.

3.1. A simple gate-limited model

In order to pursue this further, we look at a simple scenario, when $K = 2$ as shown in Fig. 4. We analyse the system by defining two chains: Chain 1, starting from $(1, 1)$ to $n, 1$, and Chain 2, starting from $(2, 2)$ to $n, 2$. For simplicity, we also assume that the size of the network buffer is infinite, the arrival rate is Poisson and the service rates are exponential.

3.2. Looking at Chain 1

Let us consider Chain 1 in Fig. 4. For Chain 1, $s = 1$ and for $n > s$ i.e. $n > 1$, we will have:

$$\lambda p_{n-1,1} = (\lambda + \mu_1) p_{n,1} \tag{5}$$

This implies that for any $n > 1$, in Chain 1:

$$p_{n,1} = \frac{\lambda}{\lambda + \mu_1} p_{n-1,1}$$

$$p_{n,1} = \left(\frac{\lambda}{\lambda + \mu_1}\right)^{n-1} p_{1,1} \tag{6}$$

Please cite this article in press as: G. Mapp et al., Exploring gate-limited analytical models for high-performance network storage servers, J. Comput. System Sci. (2010), doi:10.1016/j.jcss.2010.08.002

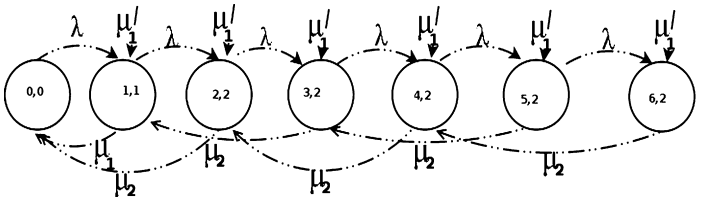

Fig. 5. Imaginary Partial Batch Model for Chain 2.

And for $n = s = 1$, we have:

$$(\lambda + \mu_1)p_{1,1} = \lambda p_{0,0} + \mu_1 p_{2,1} + \mu_2 p_{3,2} \tag{7}$$

Finally, for $n = s = 0$, i.e. $p_{0,0}$ will be:

$$\lambda p_{0,0} = \mu_1 p_{1,1} + \mu_2 p_{2,2} \tag{8}$$

3.3. Looking at Chain 2

Similarly, for Chain 2 where $s = 2$, we will derive equations for $n = s$ and $n > s$, using Fig. 4. When $n > s$, we have:

$$(\lambda + \mu_2)p_{n,2} = \lambda p_{n-1,2} + \mu_2 p_{n+2,2} + \mu_1 p_{n+1,1} \tag{9}$$

And for $n = s = 2$, we have:

$$(\lambda + \mu_2)p_{2,2} = \mu_2 p_{4,2} + \mu_1 p_{3,1} \tag{10}$$

In order to solve these equations, we need to be able to relate all the states of a chain back to the first element of that chain. So for Chain 1 there is already a relationship which is represented by Eq. (6). We now need an equation for the second chain. To do this, we first obtain an equation that has elements of Chain 2 only. We can do so by summing up the rates of the flows into point $p_{3,2}$.[1]

$$(\lambda + \mu_2)p_{3,2} = \lambda p_{2,2} + \mu_2 p_{5,2} + \mu_1 p_{4,1} \tag{11}$$

From Eq. (6), $p_{4,1}$ can be expressed as $(\frac{\lambda}{\lambda+\mu_1})^3 p_{1,1}$.
Further, from Eq. (8), $\mu_1 p_{1,1}$ can be expressed as $\lambda p_{0,0} - \mu_2 p_{2,2}$. Therefore,

$$\mu_1 p_{4,1} = (\lambda p_{0,0} - \mu_2 p_{2,2})\left(\frac{\lambda}{\lambda + \mu_1}\right)^3 \tag{12}$$

Substituting the value of $p_{4,1}$ into Eq. (11) and rearranging, we get:

$$0 = -(\lambda + \mu_2)p_{3,2} + \lambda p_{2,2} + \mu_2 p_{5,2} + (\lambda p_{0,0} - \mu_2 p_{2,2})\left(\frac{\lambda}{\lambda + \mu_1}\right)^3 \tag{13}$$

4. Solutions approach

We can now find the roots of these equations using the same technique that was used in the Partial Batch Model. Thus the state probabilities of Chain 2 for $n \geqslant 2$ can be given by:

$$p_{n,2} = r^{n-2} p_{2,2} \tag{14}$$

In order to solve this equation we imagine the second chain to be identical to a Partial Batch Model represented by Eq. (13). This is an imaginary chain as shown in Fig. 5. However, for states where $n > 2$, there is no real difference between the real or imaginary chains as Eq. (14) is valid in both scenarios. This means we can use the same approach taken in the Partial Batch Model to calculate r. Once this is done we can represent any state in the second chain by Eq. (14). In addition, using the previous equations, it will also be possible to represent $p_{0,0}$ and Chain 1 in terms of $p_{2,2}$.

Using Eq. (8), we substitute for $\lambda p_{0,0}$ in Eq. (7). In addition, we note that according to Eq. (14), $p_{3,2} = r p_{2,2}$. Rearranging, we get: $p_{1,1} = C_{1,1} p_{2,2}$ where $C_{1,1}$ is given by the equation:

[1] Similar techniques can be used for different points of Chain 2, e.g. $p_{2,2}$, $p_{4,2}$.

Please cite this article in press as: G. Mapp et al., Exploring gate-limited analytical models for high-performance network storage servers, J. Comput. System Sci. (2010), doi:10.1016/j.jcss.2010.08.002

$$C_{1,1} = \mu_2(1+r)\left(\frac{\lambda+\mu_1}{\lambda^2}\right) \tag{15}$$

By substituting for $p_{1,1}$ in Eq. (8), we can get an equation for $p_{0,0}$ in terms of $p_{2,2}$; i.e., $p_{0,0} = C_{0,0}p_{2,2}$ where $C_{0,0}$ is given by:

$$C_{0,0} = \frac{\mu_1 C_{1,1} + \mu_2}{\lambda} \tag{16}$$

4.1. Solving for $p_{2,2}$

The sum of all the state probabilities must be equal to 1. Let S_1 be the sum of the state probabilities for Chain 1 and S_2 be the sum of the state probabilities in Chain 2. So we can write:

$$p_{0,0} + S_1 + S_2 = 1 \tag{17}$$

where:

$$S_1 = \sum_{n=1}^{\infty}\left(\frac{\lambda}{(\lambda+\mu_1)}\right)^{n-1} p_{1,1} \tag{18}$$

$$S_2 = \sum_{n=2}^{\infty} r^{n-2}\, p_{2,2} \tag{19}$$

For S_1, let $m = n-1$ and substitute for $p_{1,1}$

$$S_1 = \frac{\lambda+\mu_1}{\mu_1} C_{1,1} p_{2,2} \tag{20}$$

Similarly for S_2, let $m = n-2$

$$S_2 = \frac{1}{1-r} p_{2,2} \tag{21}$$

Summing to one we get:

$$p_{2,2} = \frac{1}{C_{0,0} + \frac{\lambda+\mu_1}{\mu_1} C_{1,1} + \frac{1}{1-r}} \tag{22}$$

Using the value of $p_{2,2}$ in Eq. (22), we can find values for $p_{1,1}$ and $p_{0,0}$.
The average number of people in the queue can be expressed as:

$$L = \sum_{n=1}^{\infty} n\left(\frac{\lambda}{(\lambda+\mu_1)}\right)^{n-1} p_{1,1} + \sum_{n=2}^{\infty} n r^{n-2} p_{2,2} \tag{23}$$

We can further obtain an exact formula for L, as detailed in the section below.

4.2. Further solving for L

From Eq. (23), we first solve for the first term on the right hand side of the equation and then second term. Let $q = \frac{\lambda}{\lambda+\mu_1}$

$$\sum_{n=1}^{\infty} n\left(\frac{\lambda}{(\lambda+\mu_1)}\right)^{n-1} p_{1,1} = \sum_{n=1}^{\infty} n q^{n-1} p_{1,1} \tag{24}$$

Now, $n * q^{n-1} = \frac{d}{dq} q^n$

$$\sum_{n=1}^{\infty} \frac{d}{dq} q^n p_{1,1} = \frac{d}{dq} \sum_{n=0}^{\infty} q^n p_{1,1} \tag{25}$$

With the final result being:

$$\sum_{n=1}^{\infty} n\left(\frac{\lambda}{(\lambda+\mu_1)}\right)^{n-1} p_{1,1} = \frac{1}{(1-q)^2} p_{1,1} \tag{26}$$

Please cite this article in press as: G. Mapp et al., Exploring gate-limited analytical models for high-performance network storage servers, J. Comput. System Sci. (2010), doi:10.1016/j.jcss.2010.08.002

Now solving the second term on the right hand side of Eq. (23) we have

$$\sum_{n=2}^{\infty} nr^{n-2}p_{2,2} = \sum_{n=2}^{\infty}(n-1)r^{n-2}p_{2,2} + \sum_{n=2}^{\infty} r^{n-2}p_{2,2} \tag{27}$$

In order to present the solution of Eq. (27) in simple form, we will again solve the terms in the right hand side one by one, starting with the second term on the right hand side of Eq. (27).

$$\sum_{n=2}^{\infty} r^{n-2}p_{2,2} \tag{28}$$

Substituting $q = n-2$, we get:

$$= \sum_{q=0}^{\infty} r^{q}p_{2,2} = \frac{1}{1-r}p_{2,2} \tag{29}$$

Now, solve the first term on the right hand side of Eq. (27). By substituting $(n-1)r^{n-2} = \frac{d}{dr}r^{n-1}$ we get:

$$\sum_{n=2}^{\infty}(n-1)r^{n-2}p_{2,2} = \sum_{n=2}^{\infty}\frac{d}{dr}r^{n-1}p_{2,2} = \frac{d}{dr}\sum_{n=1}^{\infty} r^{n-1}p_{2,2} \tag{30}$$

Substituting $q = n-1$, we get:

$$= \frac{d}{dr}\sum_{n=0}^{\infty} r^{q}p_{2,2} = \frac{1}{(1-r)^2}p_{2,2} \tag{31}$$

The results expressed in Eqs. (29) and (31) showed that Eq. (27) can be expressed as:

$$\sum_{n=2}^{\infty} nr^{n-2}p_{2,2} = \left(\frac{1}{1-r} + \frac{1}{(1-r)^2}\right)p_{2,2} \tag{32}$$

$$\sum_{n=2}^{\infty} nr^{n-2}p_{2,2} = \frac{2-r}{(1-r)^2}p_{2,2} \tag{33}$$

From Eqs. (26) and (33), Eq. (23) can be expressed as:

$$L = \frac{1}{(1-q)^2}p_{1,1} + \frac{2-r}{(1-r)^2}p_{2,2} \tag{34}$$

where $q = \frac{\lambda}{\lambda+\mu_1}$ and r is the solution to the imaginary PBM chain.

The average waiting time in the demand queue, $W_d = \frac{L}{\lambda_d}$.

4.3. Comparison of results

Simulation results for $K = 2$ were obtained for different arrival rates. The simulation was done by using a discrete event-based system written in C++. More details are given in [15]. The simulation results are then compared with results for our analytical model. This is shown in Fig. 6.

The two results are quite close in value over a wide operational range. These results were used to develop techniques for prefetching and clustering using a high-performance network storage server [16,15]. It should be noted that the model is approximate as it depends on which state of the imaginary chain is used to calculate r. This is because the solution for r varies slightly depending on which state is used. The best results were obtained using the state $(2, 2)$ which, in this case, is equal to K, the maximum batch size. If $n = 2$, we get the following equation:

$$(\lambda + \mu_2)p_{2,2} = \lambda p_{1,1} + \mu_2 p_{4,2} + \mu_1 p_{3,1} \tag{35}$$

Note that $p_{1,1}$ comes from the imaginary PBM and not from the gate-limited service model.

Please cite this article in press as: G. Mapp et al., Exploring gate-limited analytical models for high-performance network storage servers, J. Comput. System Sci. (2010), doi:10.1016/j.jcss.2010.08.002

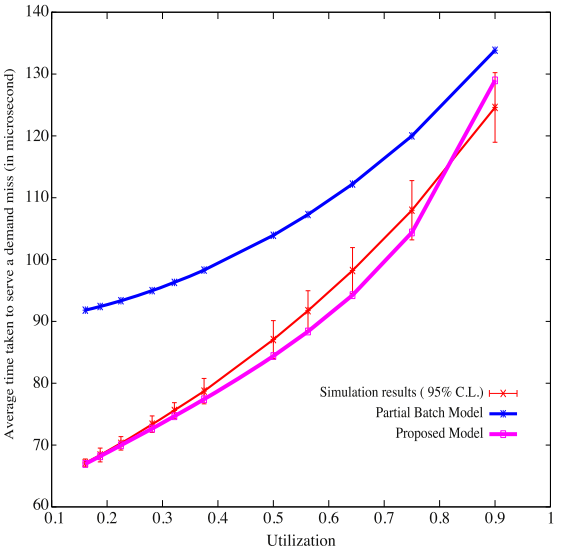

Fig. 6. Results for simple gate-limited model with $K = 2$.

4.4. Towards a general solution

In this section, we seek to extend the method used for $K = 2$ to a general value of K. So a gate-limited model, where K is equal to the maximum number of requests that can be served at any moment, can be represented by a gate-limited model of K series or chains. Furthermore, if we represent a given chain by m, we can express the average number of requests in that chain, L_m, in terms of the first element of that chain, $p_{m,m}$. For $m < K$, this sum for that chain is given by:

$$L_m = \sum_{n=m}^{\infty} n \left(\frac{\lambda}{(\lambda + \mu_m)} \right)^{n-m} p_{m,m} \tag{36}$$

Expanding, we get:

$$L_m = \sum_{n=m}^{\infty} (n - (m-1)) \left(\frac{\lambda}{(\lambda + \mu_m)} \right)^{n-m} p_{m,m} + (m-1) \sum_{n=m}^{\infty} \left(\frac{\lambda}{(\lambda + \mu_m)} \right)^{n-m} p_{m,m} \tag{37}$$

Using the same technique as above and by letting $r_m = \frac{\lambda}{\lambda + \mu_m}$, the first term can be expressed as:

$$\sum_{n=m}^{\infty} (n - (m-1)) r_m^{n-m} p_{m,m} = \sum_{n=m}^{\infty} \frac{d}{dr} r_m^{n-(m-1)} p_{m,m} \tag{38}$$

Rearranging, we get:

$$\sum_{n=m}^{\infty} \frac{d}{dr} r_m^{n-(m-1)} p_{m,m} = \frac{d}{dr} \sum_{n=m-1}^{\infty} r_m^{n-(m-1)} p_{m,m} \tag{39}$$

Let $p = n - m + 1$

$$= \frac{d}{dr} \sum_{p=0}^{\infty} r_m^p p_{m,m} = \frac{1}{(1 - r_m)^2} p_{m,m} \tag{40}$$

The second term:

$$(m-1) \sum_{n=m}^{\infty} \left(\frac{\lambda}{(\lambda + \mu_m)} \right)^{n-m} p_{m,m} = (m-1) \sum_{n=m}^{\infty} r_m^{n-m} p_{m,m} \tag{41}$$

Let $q = n - m$;

$$= (m-1) \sum_{q=0}^{\infty} r_m^q p_{m,m} = (m-1) \frac{1}{1 - r_m} p_{m,m} \tag{42}$$

and thus we get the sum:

$$L_m = \frac{m-(m-1)*r_m}{(1-r_m)^2} p_{m,m} \tag{43}$$

$$L = \sum_{m=1}^{K} L_m = \sum_{m=1}^{K} \frac{m-(m-1)*r_m}{(1-r_m)^2} p_{m,m} \tag{44}$$

For $m < K$,

$$r_m = \frac{\lambda}{\lambda+\mu_m} \tag{45}$$

For $m = K$, we use the imaginary PBM technique to solve for r_K. Furthermore, for $m < K$, we can sum the probabilities in each chain,

$$S_m = \sum_{n=m}^{\infty} \left(\frac{\lambda}{(\lambda+\mu_m)} \right)^{n-m} p_{m,m} \tag{46}$$

Let $q = n - m$:

$$S_m = \sum_{q=0}^{\infty} \left(\frac{\lambda}{(\lambda+\mu_m)} \right)^{q} p_{m,m}$$

$$S_m = \frac{\lambda+\mu_m}{\mu_m} p_{m,m} \tag{47}$$

If we let $p_{m,m} = C_{m,m} p_{K,K}$, we can express $p_{K,K}$ as:

$$p_{K,K} = \frac{1}{C_{0,0} + \sum_{m=1}^{m=K-1} \frac{\lambda+\mu_m}{\mu_m} C_{m,m} + \frac{1}{1-r_k}} \tag{48}$$

So to solve for any value of K we need to find the value of $C_{m,m}$ and we can do so using the equations for the states of $p_{m,m}$ in our model. For $K = 2$, these equations are Eqs. (7), (8) and (9). However, this effort shows that it is possible to get fairly accurate waiting time results over a wide operational range based on this analytical model. Work is continuing on evaluating higher orders of K.

5. Designing a high-performance storage server – the issues

5.1. Motivation

The continuous increase in network and CPU speeds is fostering the development of a new class of network applications based on streaming, for example, YouTube and BBC iPlayer. Unlike traditional client-server applications which use best-effort transport mechanisms, these new applications require QoS support to deliver acceptable performance. In addition, since the access patterns of these applications are highly sequential, prefetching techniques, in which blocks of data are brought into memory before they are needed, can be used to maintain a steady access rate and so prevent stalling.

However, for prefetching to be effective, it must be relatively inexpensive, such that the cost of bringing in additional blocks is small compared with fetching these blocks when they are needed by the application. One of the ways to ensure that prefetching is cheap is to use clustering in which the cost-per-block is kept low as several blocks are fetched at the same time. It should be expected that high-performance network-based servers would make use of clustering.

Storage systems are also expected to support demand misses which occur when a block of data is needed by a program to continue its execution. Demand misses must therefore be promptly satisfied or else applications will experience severe execution delays. So it is necessary to balance the demands of prefetching using clustering and the need to promptly satisfy demand misses. In order to explore the design of high-performance storage servers, a Network Memory Server (NMS) has been developed at Middlesex University.

5.2. Characteristics of the Network Memory Server

The NMS has been designed with a number of key goals. Firstly, the actions of the NMS are regarded as atomic by its clients. This means that calls to the NMS either succeed completely or fail completely. Secondly, the NMS is stateless. It stores no previous interactions between client and server. Finally, the NMS is designed to be mobile and can move around different computing environments as the user moves around. These properties tend to indicate that it is reasonable to assume a Poisson-distribution for the arrival rate and the service rate can also assumed to be exponential since the system is memoryless with regard to previous transactions. A more detailed presentation of the architecture is given in [17].

Please cite this article in press as: G. Mapp et al., Exploring gate-limited analytical models for high-performance network storage servers, J. Comput. System Sci. (2010), doi:10.1016/j.jcss.2010.08.002

5.3. Investigating clustering on the Network Memory Server

We first investigated clustering in which several blocks are fetched simultaneously using the NMS for a lightly-loaded network and found that this can be represented by a simple equation given by:

$$T_{net}(y) = Lat_{net} + Cy \tag{49}$$

where T_{net} is the time taken to fetch y blocks over the network, where Lat_{net} is a latency cost, i.e., the cost of going up and down the network protocol stacks on the client and server as well as the cost of transmitting and receiving packets over the network. For very fast networks and fast processors, Lat_{net} can be very small. Finally, C is the per-block cost. So this is the cost of finding the block in memory and copying the data to and from the network buffer. For the NMS, C has been measured and has a value of around 30 μs on a 1 Gbps network in a Linux environment.

5.4. Analysing the constraints

In this section, we derive equations that represent the constraints that must be satisfied in order to meet the requirements of prefetching and demand misses. In terms of prefetching we can represent the time to consume y block as:

$$T_{process}(y) = T_{cpu} * y \tag{50}$$

where $T_{process}$ is the time taken to consume y blocks and T_{cpu} is the time taken to consume one block. In order to prevent stalling for streaming applications, the time taken to fetch y blocks must be less than or equal to the time taken to consume these blocks so:

$$Lat_{net} + Cy \leqslant T_{cpu} * y \tag{51}$$

Next we look at demand misses. Since demand misses must be satisfied promptly, we would like to ensure that demand misses over the network experience better performance compared to local disk access in order to justify the use of network storage. So we can represent this by the equation:

$$T_{fetch(d)} + T_{wait} \leqslant T_{disk} \tag{52}$$

5.5. Investigating prefetching strategies

In order to continue our design, it is necessary to look at different prefetching strategies and decide which one would be most suitable for the network memory environment. There are basically three prefetching strategies. The first is called **Aggressive** prefetching. With this strategy, prefetching is done at the earliest opportunity, i.e. whenever memory and network resources are available. This strategy was proposed in [18] and is based on the fact that modern computer systems now have large amounts of memory, faster processors and can utilise higher network bandwidths.

The second strategy is called **Just-In-Time** or JIT prefetching. With this strategy, prefetching is done such that the block is made available to the system just before it is required by the application. This idea was explored in a disk-centric environment with parallel disks in [19]. The third prefetching strategy is called **Prefetch-on-Demand** or PonD. With this strategy prefetching only occurs when demand misses are being satisfied. This is also known as conservative prefetching and was explored for disk storage by Pei Cao in [20].

In order to decide which prefetching strategy was appropriate, a simulation of the effect of each prefetching strategy on the average waiting time of demand misses was developed. The results are shown in Fig. 7 and clearly indicate that the PonD strategy would be the best strategy over the network. Hence we concentrate on using PonD.

However, the use of PonD means that we must also modify our prefetching and demand-miss constaints because we are fetching both prefetch and demand-miss blocks at the same time. So if p is the number of prefetch blocks and d is the number of demand-miss blocks, then we can express the prefetch constraint in the PonD environment as:

$$Lat_{net} + C(p + d) \leqslant T_{cpu} * p \tag{53}$$

Similarly we can represent the demand-miss constraint in the PonD environment as:

$$Lat_{net} + C(p + d) + T_{wait} \leqslant T_{disk} \tag{54}$$

where the fetch time for d blocks is $L + C(p + d)$ and T_{wait} is the average waiting time in the demand queue.

The environment that we are trying to support can be represented by the queueing model shown in Fig. 8. A more detailed description of this environment is given in [21]. There are 2 queues, a demand queue and a prefetch queue with arrival rates λ_d and λ_p respectively. Requests from both queues are clustered into a buffer, b, which is then sent over the network to a high-performance storage server that returns the requested blocks.

In order to simplify the analysis, we first make a key assumption. Since we are in complete control of the prefetch queue, we can determine the rate of prefetching by starting and stopping streams. Hence λ_p is not a random variable. In addition,

Please cite this article in press as: G. Mapp et al., Exploring gate-limited analytical models for high-performance network storage servers, J. Comput. System Sci. (2010), doi:10.1016/j.jcss.2010.08.002

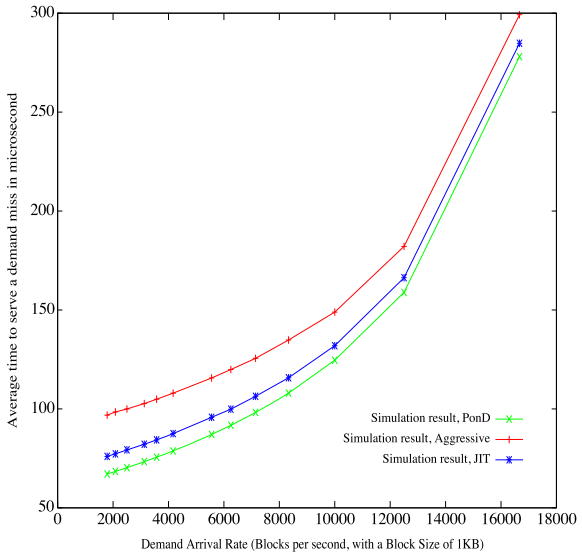

Fig. 7. Showing the effect of prefetching strategies on demand misses.

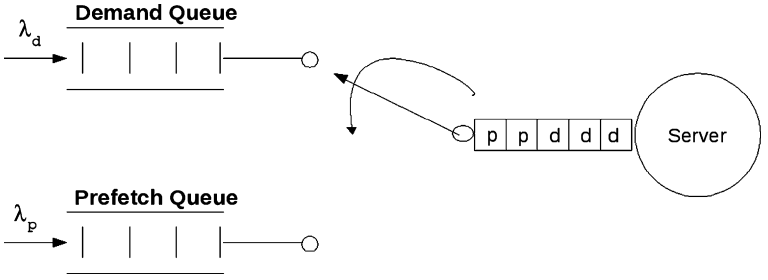

Fig. 8. A model showing prefetch and demand queues.

for prefetching we are only interested in knowing how many blocks denoted by the variable, p, to prefetch at any instant, such that the prefetch rate is always more than the rate at these blocks are consumed, hence preventing stalling. In this analysis the demand miss rate is truly random. Given these observations we can therefore reduce the two-queue system to a single server system based on demand misses. However, the service time for d demand miss blocks will also include the time to prefetch p blocks as well. This therefore corresponds to the gate-limited analysis done earlier in the paper.

6. Developing an autonomous algorithm

6.1. Exploring an operational space

To develop an autonomous system, we need to explore an operational space. This is shown in Fig. 9. This region can be viewed as a 3D object. The 3 axes of the space consist of the prefetch rate, represented as the number of blocks, p, that must be prefetched to maintain a given rate (x-axis), the demand miss rate, λ_d, (y-axis) and the average waiting time experienced by demand misses, W_d (z-axis).

However, it should be observed that since we cannot control the demand miss rate, λ_d, it is necessary to explore the operational space as 2D slices, each slice representing a particular demand miss rate. For each miss rate we obtain a graph with the average waiting time (y-axis) and the number of prefetch blocks being fetched for different values of demand miss blocks, d. So for each demand miss we apply the two constraints in Eqs. (53) and (54). This is shown in Fig. 10. In this figure, we show that average waiting times as indicated by the simulation. Note: the analytical results could have also been used. Simulation results are used as they were easier to generate.

For the disk constraint, T_{disk}, the disk access time was set at a value which modern SATA hard drive could achieve, so it was set at 7800 μs. Once the average waiting time is less than T_{disk}, then the disk constraint is satisfied. In this figure, this constraint is satisfied for $p \leqslant 17$ blocks. For the prefetch constraint, the time taken to consume p blocks is $T_{process} = T_{cpu} * p$. T_{cpu} is different for the different video formats. For MPEG, T_{cpu} is 400 μs while for High-Definition (HD) the consume rate is 200 μs. So the prefetch constraint can be shown as a line with gradient T_{cpu}. Hence once the time taken to consume the blocks is above the line taken to fetch the blocks (i.e. the average access time), then the prefetch constraint is satisfied.

Please cite this article in press as: G. Mapp et al., Exploring gate-limited analytical models for high-performance network storage servers, J. Comput. System Sci. (2010), doi:10.1016/j.jcss.2010.08.002

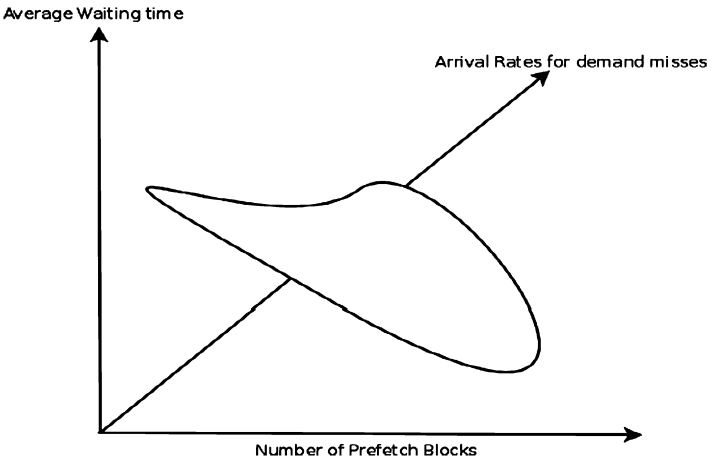

Fig. 9. Exploring the operational space.

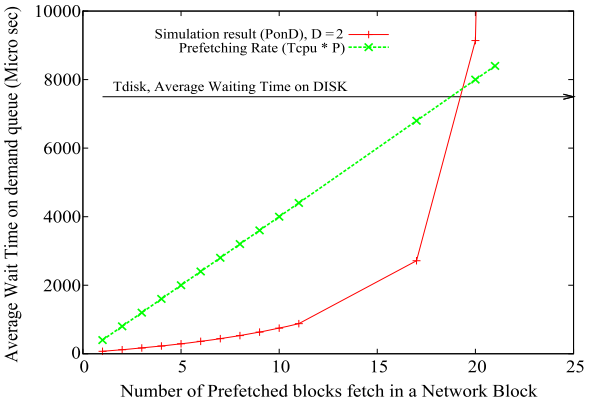

Fig. 10. Showing the two constraints and the optimal operational point for $d = 2$ and an inter-arrival time of 350 μs.

Hence from Fig. 10, the prefetch constraint is satisfied for $p \leqslant 19$ blocks. So if $p \leqslant 17$ blocks then both the storage and prefetch constraints are satisfied.

6.2. Optimal operational points

Since the PonD strategy is being used, it is necessary to try and bring in the maximum number of prefetch blocks, denoted by P_{opt}, on every network access while satisfying the two constraints. From the observations above, this can be expressed as:

$$T_{cpu} * P_{opt} = T_{disk} = W_d \tag{55}$$

This point is called an optimal operational point. There is one optimal operational point for one value of d for every inter-arrival time which is called the *opt_time*. This is shown in Fig. 11. In addition, if we know P_{opt} for a given value of d, then we can define an optimal consume time, T_{opt}, which is the minimum consume time for that optimal operational point. If the consume time is less than T_{opt} (smaller gradient), then a smaller amount or sub_optimal value of p will be required.

Hence in order to support a smaller consumer time, i.e., a faster consume rate, using optimal operational points, it is necessary to move to higher values of d as shown in Fig. 12.

If we are able to measure λ_d, and we know the consume rate for streaming applications, then it is possible to find out the optimal value P_{opt} and the value of d which should be used to fetch blocks from network storage.

An autonomous algorithm was developed using 20 different arrival rates for demand misses. For each arrival rate, P_{opt} and T_{opt} were calculated for values of d from 1 to 7 blocks. These values were placed in a database, which was then incorporated into a systems simulation to test whether or not the system would perform in response to changes in the arrival rate or changes in the prefetch rate.

Please cite this article in press as: G. Mapp et al., Exploring gate-limited analytical models for high-performance network storage servers, J. Comput. System Sci. (2010), doi:10.1016/j.jcss.2010.08.002

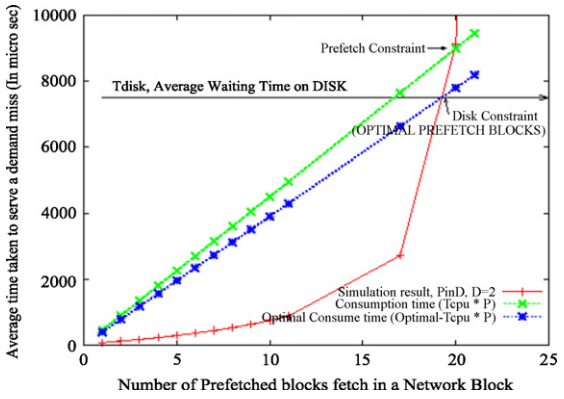

Fig. 11. Showing the two constraints and the optimal operational point for $d = 2$ and an inter-arrival time of 450 µs.

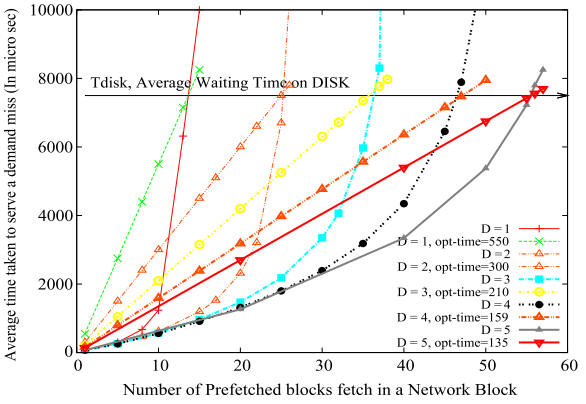

Fig. 12. Showing optimal operational points at different values of d and an inter-arrival time of 450 µs.

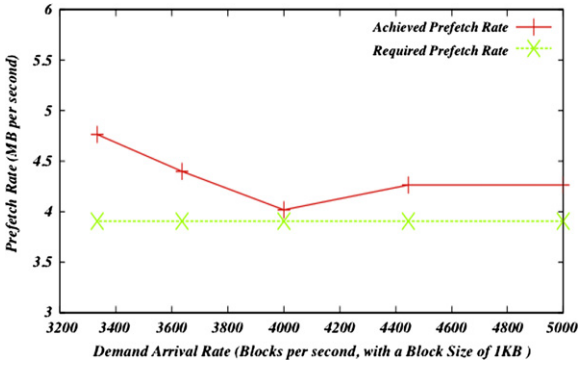

Fig. 13. Showing that the prefetch rate is maintained as the demand-miss rate is varied.

Fig. 13 shows the autonomous algorithm maintains the prefetch rate at just above or equal to the required prefetch rate as the demand-miss rate is varied while Fig. 14 shows how the disk constraint, i.e., $W_d \leqslant T_{disk}$, is met when the prefetch rate is varied.

6.3. Development of an Experimental File System

In order to evaluate this algorithm on a real system, an Experimental File System (EFS) was set up containing two major directories: the SEQUENTIAL and DEMAND directories. Files in the SEQUENTIAL directory were prefetched while files in the DEMAND directory were satisfied using demand misses. The design is shown in Fig. 15. Fig. 16 shows how data is read by the application using the file system as well as the multiread call to the NMS.

The system was then used to play a number of MPEG videos. The demand miss rate was measured and the consume time of the video was 400 µs per block. This was used to select P_{opt} and d, the number of demand miss blocks that should

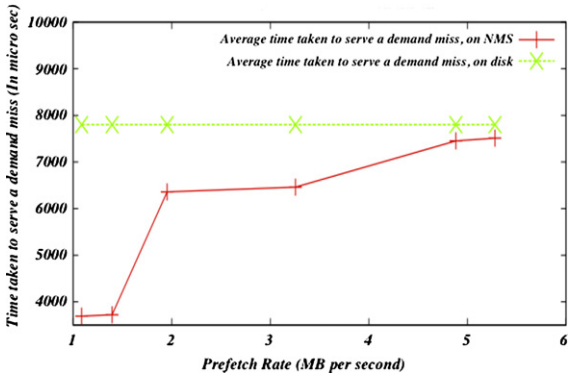

Fig. 14. Showing the storage constraint is met as the prefetch rate is varied.

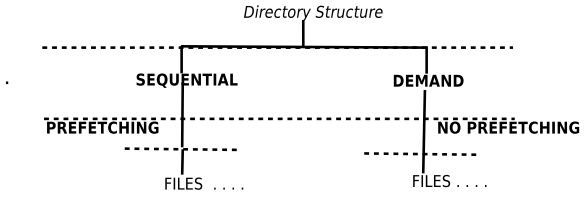

Fig. 15. File system design.

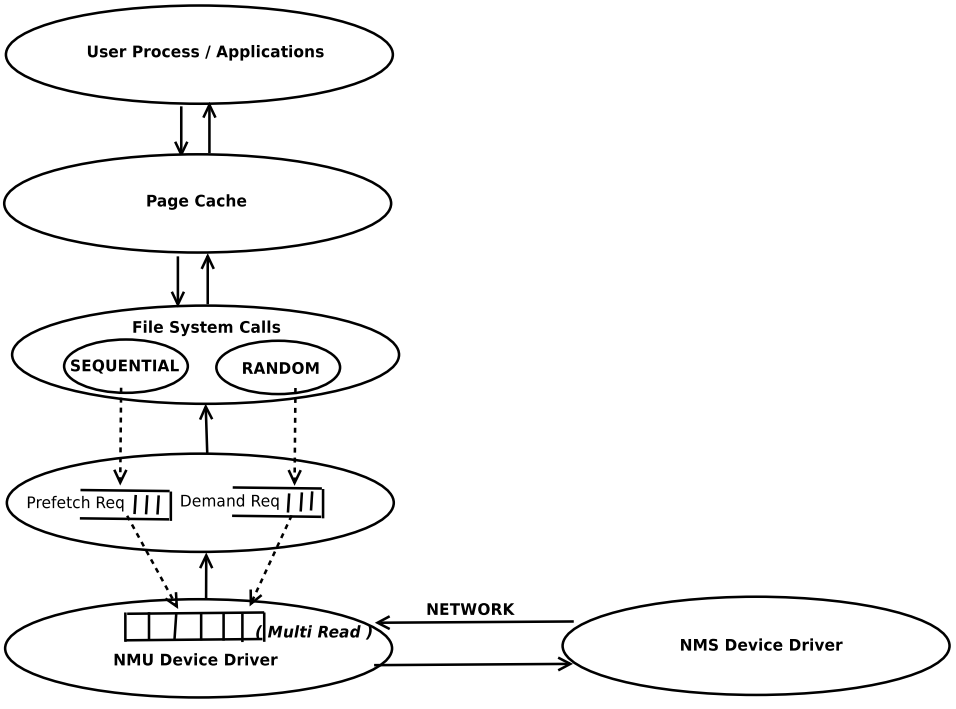

Fig. 16. Overall system design.

be fetched in the next network access. This was done using the multiread call. The video blocks were therefore prefetched and hence the video did not stall. The output of the system is shown in Fig. 17. On the left of the screen, we see the system measuring the arrival rate of demand misses and using the MPEG consume time, selects P_{opt} and d. The video being viewed is shown on the right of the frame. Preliminary results showed that the algorithm worked admirably on this prototype system.

Please cite this article in press as: G. Mapp et al., Exploring gate-limited analytical models for high-performance network storage servers, J. Comput. System Sci. (2010), doi:10.1016/j.jcss.2010.08.002

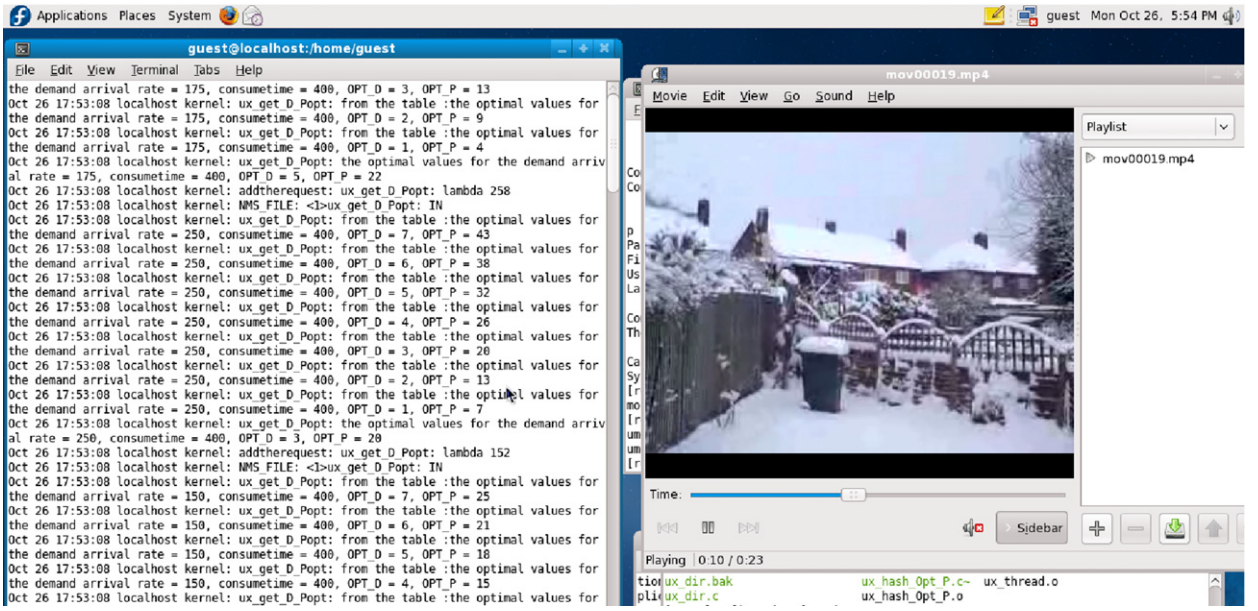

Fig. 17. The output of the Experimental File System.

7. Conclusions

In this paper, an analytical model for gate-limited service was explored and applied to a high-performance server doing prefetching using clustering techniques. The model shows a high level of accuracy for simple gate-limited service at operational loads. Using the concept of optimal operational points, an algorithm was developed and tested on a real system using an Experimental File System (EFS).

References

[1] J. Sykes, Simplified analysis of alternating-priority queueing model with setup times, Oper. Res. 18 (6) (November–December 1970) 1182–1192.
[2] Y. Aminetzah, An exact approach to the polling system, PhD thesis, Dept. of Electrical Engineering, McGill Univ., Montreal, Quebec, 1975.
[3] R.B. Cooper, Introduction to Queueing Theory, 2d ed., Elsevier, North-Holland, New York, 1981.
[4] O. Gemikonakli, T. Do, R. Chakka, E. Ever, Numerical solution to the performability of a multiprocessor system with reconfiguration and rebooting delays, in: Proceedings of ECMS 2005, June 2005, pp. 766–773.
[5] Q. Zeng, K. Mukumoto, A. Fukuda, Performance analysis of mobile cellular radio system with priority reservation and handoff procedures, in: Proceedings of IEEE VTC 1994, June 1994, pp. 1829–1833.
[6] K. Trivedi, X. Ma, Performability analysis of wireless cellular networks, in: Proceedings of Symposium on Performance Evaluation of Computer and Telecommunication System (SPECTS 2002), San Diego, USA, July 2002.
[7] H. Takagi, Queueing Analysis of Polling Models, ACM, 1988.
[8] K. Leung, Cyclic-service systems with probabilistically-limited service, IEEE J. Selected Areas in Communications 9 (2) (February 1991).
[9] K. Chang, D. Sandhu, Pseudo-conservation laws in cyclic-service systems with a class of limited service policies, Ann. Oper. Res. 35 (3) (June 1992).
[10] R. Dittmann, F. Hubner, Discrete-time analysis of a cyclic service system with gated limited service, Institute of Computer Science, University of Wurzburg, Tech. Rep., June 1993.
[11] M. van Vuuren, E. Winands, Interactive approximation of k-limited polling systems, Technische Universiteit Eindhoven, Tech. Rep., May 2006.
[12] D. Gross, C.M. Harris, Fundamentals of Queueing Theory, Wiley Ser. Probab. Stat., Wiley–Interscience, February 1998.
[13] W.J. Stewart, Probability, Markov Chains, Queues, and Simulation, Princeton University Press, 2009.
[14] W.J. Stewart, Introduction to the Numerical Solution of Markov Models, Princeton University Press, 1994.
[15] D.N. Thakker, Prefetching and clustering techniques for network based storage, School of Engineering and Information Sciences, Middlesex University, PhD thesis, May 2010.
[16] G. Mapp, D. Thakker, O. Gemikonakli, Exploring gate-limited analytical models for high performance network storage servers, in: Proceedings of the 3rd International Workshop on Performance Modeling and Evaluation in Computer and Telecommunication Networks (PMECT 2009), San Francisco, USA, August 2009.
[17] G. Mapp, D. Thakker, D. Silcott, The design of a storage architecture for mobile heterogeneous devices, in: ICNS2007, vol. 0, 2007, p. 41.
[18] A.E. Papathanasiou, M.L. Scott, Aggressive prefetching: an idea whose time has come, in: HOTOS '05: Proceedings of the 10th Conference on Hot Topics in Operating Systems, USENIX Association, Berkeley, CA, USA, 2005, pp. 1–6.
[19] R.H. Patterson, G.A. Gibson, E. Ginting, D. Stodolsky, J. Zelenka, Informed prefetching and caching, in: SOSP '95: Proceedings of the Fifteenth ACM Symposium on Operating Systems Principles, ACM, New York, NY, USA, 1995, pp. 79–95.
[20] P. Cao, Application-controlled file caching and prefetching, PhD dissertation, Princeton, NJ, USA, 1996.
[21] D. Thakker, G. Mapp, Clustering and prefetching techniques for network-based storage systems, in: Eighteenth Annual Workshop on Information Technologies and Systems (WITS), Paris, France, December 13–14, 2008.

Please cite this article in press as: G. Mapp et al., Exploring gate-limited analytical models for high-performance network storage servers, J. Comput. System Sci. (2010), doi:10.1016/j.jcss.2010.08.002